O.s Lab Manual

1. Practicing of Basic UNIX Commands.

Aim: To Practice Basic UNIX Commands

UNIX COMMANDS:

a) date

–used to check the date and time

Syn:$date

Format	Purpose	Example	Result
+%m	To display only month	$date+%m	06
+%h	To display month name	$date+%h	June
+%d	To display day of month	$date+%d	O1
+%y	To display last two digits of years	$date+%y	09
+%H	To display hours	$date+%H	10
+%M	To display minutes	$date+%M	45
+%S	To display seconds	$date+%S	55

b) cal

–used to display the calendarSyn:$cal 2 2009

c) echo

–used to print the message on the screen.

Syn:$echo "text"

d) ls

–used to list the files. Your files are kept in a directory.

Syn:$lsls–s

All files (include files with prefix) ls–l Lodetai (provide file statistics) ls–t Order by creation time

ls– u Sort by access time (or show when last accessed together with –l)ls–s Order by size

ls–r Reverse order

ls–f Mark directories with /,executable with* , symbolic links with @, local sockets with =,named pipes(FIFOs)with

ls–s Show file size

ls– h" Human Readable", show file size in Kilo Bytes & Mega Bytes (h can be used together with –l or)

FILE MANIPULATION COMMANDS

Command : Cat
Purpose : It is used to display the contents of the file as well as used to create a new file.
Syntax : cat <file name >
Example : $ cat devi

Command : More

Purpose : It is used to display the contents of the file on the screen at a time.
Syntax : more <file name >
Example : $ more devi

Command : Wc
Purpose : It is used to count the number of lines ,words and characters in a file or group of files.
Syntax : wc [options] <file name >
Example : $ wc –l devi

Command : File
Purpose : It is used to determine the type of the file.
Syntax : file <file name >
Example : $ file devi

Command : Spell
Purpose : It is used to find the spelling errors in the file.
Syntax : spell [options] <file name >
Example : $ spell -b devi

Command : Split
Purpose : It is used to split the given file into smaller pieces of given size.
Syntax : split –size <file name > < splitted file name >
Example : $ split –2 devi de

Command : CP
Purpose : It is used to copy one or more files.
Syntax : cat <source file name > <destination file name>
Example : $ cp devi latha

Command : MV
Purpose : It is used to move a file within a directory with different names and also used to move a file to different directory with its original name.
Syntax : mv <source file name > <destination file name>
Example : $ mv devi jeya

Command : RM
Purpose : It is used to remove a file from the disk.
Syntax : rm <file name >
Example : $ rm devi

1. a) GENERAL PURPOSE COMMANDS

Command : Banner
Purpose : It is used to display its argument in large letters.
Syntax : banner < string >
Example : $ banner BOOM

Command : Who
Purpose : It is used to get the information about all the users currently working in the system.
Syntax : who
Example : $ who

Command : Who am i
Purpose : It is used to know in which terminal the user is currently logged on.
Syntax : who am i
Example : $ who am I

Command : Tput
Purpose : It is used to manipulate the screen.
Syntax : tput < attributes >
Example : $ tput rmso

Command : Uname
Purpose : It is used to display the details about the OS in which we are working.
Syntax : uname [options]
Example : $ uname –n

Command : Tty
Purpose : It is used to know the terminal name on which we work.
Syntax : tty
Example : $ tty

Command : Pwd
Purpose : It is used to display the absolute pathname of current working directory.
Syntax : pwd
Example : $ pwd

1. b) COMMAND GROUPING & FILTER COMMANDS

Command : Head
Purpose : It is used to display the top portion of the file.
Syntax : head [options] <file name>
Example : $ head -5 devi

Command : Tail
Purpose : It is used to display the bottom portion of the file.
Syntax : tail [options] <file name >
Example : $ tail –5 devi

Command : Pr
Purpose : It is used to display the contents of the file by separating them into pages and each page begins with the header information.
Syntax : pr [options] <file name >
Example : $ pr devi

Command : Cut
Purpose : It is used to extract selected fields or columns from each line of one or more files and display them on the standard output device.
Syntax : cut [options] <file name >
Example : $ cut –c5 devi

Command : Paste
Purpose : It concatenates the line from each input file column by column with tab characters in between them.
Syntax : paste [options] <file name >
Example : $ paste f1 f2

Command : Join
Purpose : It is used to extracts common lines from two sorted files and there should be the common field in both file.
Syntax : join [options] <file name1 ><file name 2>
Example : $ join –a1 f1 f2

Command : Uniq
Purpose : It compares adjacent lines in the file and displays the output by eliminating duplicate adjacent lines in it.
Syntax : uniq [options] <file name >
Example : $ uniq -c devi

Command : Sort
Purpose : It sorts one or more files based on ASCII sequence and also to merge the file.
Syntax : sort [options] <file name >
Example : $ sort -r devi

Command : Nl
Purpose : It is used to add the line numbers to the file.
Syntax : nl [options] [filename]
Example : $ nl devi

Command : Tr
Purpose : It is used to translate or delete a character or a string from the standard input to produce the required output.
Syntax : tr [options] <string1> <string2>
Example : $ tr –t „a" „b" < devi>

Command : Tee
Purpose : It is used to read the contents from standard input or from output of another command and reproduces the output to boyh in standard output and direct into output to one or more files.
Syntax : tee [options] <file name >
Example : $ tee date dat.txt

Command : grep
Purpose : It is used to search the specified pattern from one or more files.
Syntax : grep [options] <pattern> <file name >
Example : $ grep “anand” devi

1. c) DIRECTORY COMMANDS AND PROCESS MANAGEMENT COMMANDS

Aim

To Study about directory handling and Process Management Commands

Command : mkdir
Purpose : It is used to create new directory or more than one directory.
Syntax : mkdir <directory name >
Example : $ mkdir riya

Command : rmdir
Purpose : It is used to remove the directory if it is empty.
Syntax : rmdir <directory name >
Example : $ rmdir riya

Command : cd
Purpose : It is used to change the control from one working directory to another specified directory.
Syntax : cd <directory name >
Example : $ cd riya

Command : cd ..
Purpose : It is used to quit from current directory and move to the previous directory.
Syntax : cd ..
Example : $ cd ..

Process Commands

Command : echo $$
Purpose : It is used to display the process number of the current shell.
Syntax : echo $$
Example : $ echo $$

Command : ps
Purpose : It is used to display the attributes of a process.
Syntax : ps
Example : $ ps
$ ps –f (Display the ancestry of a process)
$ ps –u (Display the activities of a user)
$ ps –a (Lists processes of all users but not the system processes)

Command :&
Purpose : It is shell operator which is used to run a process in the background.
Syntax : <command> &
Example : $ sort emp.txt &

Command : nohup
Purpose : It permits the execution of the process even after the user has logged out.
Syntax : nohup <command>
Example : $ nohup sort emp.txt (result is available on nohup.out)

Command : kill
Purpose : It is used to terminate the process.
Syntax : kill <PID>
Example : $ kill 105

Command : kill $!
Purpose : $! is the system variable which stores the process id of the last background job. The command kill $! is used to kill the last process.
Syntax : kill $!
Example : $ kill $!

Command : at
Purpose : It is used to execute the process at the time specified.
Syntax : echo <time>
Example : $ at 14:08 (or) $ at 3 PM (or) $ at 4 :50 AM

2. Write programs using the following UNIX operating system calls

Fork, exec, getpid, exit, wait, close, stat, opendir and readdir

Aim: To execute various given UNIX commands.

Program-1: Program for system calls of unix operating systems (opendir, readdir, closedir)

```
#include<stdio.h>
#include<dirent.h>
struct dirent *dptr;
int main(int argc, char *argv[])
{
char buff[100];
DIR *dirp;
printf("\n\n ENTER DIRECTORY NAME");
scanf("%s", buff);
if((dirp=opendir(buff))==NULL)
{
printf("The given directory does not exist");exit(1);
}
while(dptr=readdir(dirp))
{
printf("%s\n",dptr->d_name);
}
closedir(dirp);
}
```

Program-2: program for system calls of unix operating system (fork, getpid, exit)

```
#include<stdio.h>
#include<unistd.h>
main()
{
int pid,pid1,pid2;
pid=fork();
if(pid==-1)
{
printf("ERROR IN PROCESS CREATION \n");
exit(1);
}
if(pid!=0)
{
pid1=getpid();
printf("\n the parent process ID is %d\n", pid1);
}
else
{
pid2=getpid();
printf("\n the child process ID is %d\n", pid2);
}
}
```

3. Simulate UNIX commands like cp, ls, grep, etc.,

Aim: To simulate UNIX commands like cp, ls, grep

Program:

Program-1: .Program for simulation of cp unix command.

```
#include<fcntl.h>
#include<unistd.h>
#include<stdio.h>
main(int argc,char *argv[])
{
FILE *fp;char ch; int sc=0;
fp=fopen(argv[1],"r");
if(fp==NULL)
  printf("unable to open a file",argv[1]);
  else
{
while(!feof(fp))
{
ch=fgetc(fp);
if(ch==' ')
sc++;
}
printf("no of spaces %d",sc);
printf("\n");
fclose(fp);
}
}
```

Program-2: program for simulation of ls unix command.

```
#include<stdio.h>
#include<dirent.h>
void main(int argc, char **argv)
{
DIR *dp;
struct dirent *link;
dp=opendir(argv[1]);
printf("\n contents of the directory %s are \n", argv[1]);
while((link=readdir(dp))!=0)
printf("%s",link->d_name);
closedir(dp);
}
```

Program-3: program for simulation of grep unix command.

```
#include<stdio.h>
#include<string.h>
#define max 1024
void usage()
{
printf("usage:\t. /a.out filename word \n ");
}
int main(int argc, char *argv[])
```

```
{
FILE *fp;
char fline[max];
char *newline;
int count=0;
int occurrences=0;
if(argc!=3)
{
usage();
exit(1);
}
if(!(fp=fopen(argv[1],"r")))
{
printf("grep: couldnot open file : %s \n",argv[1]);
exit(1);
}
while(fgets(fline,max,fp)!=NULL)
{
count++;
if(newline=strchr(fline, ,,\n"))
*newline="\0";
if(strstr(fline,argv[2])!=NULL)
{
printf("%s: %d %s \n", argv[1],count, fline);
14 | P a g e
occurrences++;
}
}
}
```

Aim : Write a C program to simulate the following non-preemptive CPU scheduling algorithms to find turnaround time and waiting time for the above problem.
a) FCFS b) SJF c) Round Robin d) Priority

a)FCFS CPU SCHEDULING ALGORITHM *:*

PROGRAM:

```
#include<stdio.h>
main()
{
	int bt[20], wt[20], tat[20], i, n;
	float wtavg, tatavg;
	printf("\nEnter the number of processes -- ");
	scanf("%d", &n);
	for(i=0;i<n;i++)
	{
		printf("\nEnter Burst Time for Process %d -- ", i);
		scanf("%d", &bt[i]);
	}
	wt[0] = wtavg = 0;
	tat[0] = tatavg = bt[0];
	for(i=1;i<n;i++)
	{
		wt[i] = wt[i-1] +bt[i-1];
		tat[i] = tat[i-1] +bt[i];
		wtavg = wtavg + wt[i];
		tatavg = tatavg + tat[i];
	}
	printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND
	TIME\n");
	for(i=0;i<n;i++)
		printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
	printf("\nAverage Waiting Time -- %f", wtavg/n);
	printf("\nAverage Turnaround Time -- %f", tatavg/n);
}
```

INPUT:

Enter the number of processes -- 3
Enter Burst Time for Process 0 -- 24
Enter Burst Time for Process 1 -- 3
Enter Burst Time for Process 2 -- 3

OUTPUT:

PROCESS	BURST TIME	WAITING TIME	TURNAROUND TIME
P0	24	0	24
P1	3	24	27
P2	3	27	30

Average Waiting Time-- 17.000000
Average Turnaround Time -- 27.000000

b)SJF CPU SCHEDULING ALGORITHM :

PROGRAM:

```
#include<stdio.h>
main()
{
        int p[20], bt[20], wt[20], tat[20], i, k, n, temp;
        float wtavg, tatavg;
        printf("\nEnter the number of processes -- ");
        scanf("%d", &n);
        for(i=0;i<n;i++)
        {
                p[i]=i;
                printf("Enter Burst Time for Process %d -- ", i);
                scanf("%d", &bt[i]);
        }
        for(i=0;i<n;i++)
                for(k=i+1;k<n;k++)
                        if(bt[i]>bt[k])
                        {
                        temp=bt[i];
```

```
                        bt[i]=bt[k];
                        bt[k]=temp;
                        temp=p[i];
                        p[i]=p[k];
                        p[k]=temp;
                        }
        wt[0] = wtavg = 0;
        tat[0] = tatavg = bt[0];
        for(i=1;i<n;i++)
        {
                wt[i] = wt[i-1] +bt[i-1];
                tat[i] = tat[i-1] +bt[i];
                wtavg = wtavg + wt[i];
                tatavg = tatavg + tat[i];
        }
        printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND
        TIME\n");
        for(i=0;i<n;i++)
                printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
                printf("\nAverage Waiting Time -- %f", wtavg/n);
        printf("\nAverage Turnaround Time -- %f", tatavg/n);
}
```

INPUT:

```
Enter the number of processes -- 4
Enter Burst Time for Process 0 -- 6
Enter Burst Time for Process 1 -- 8
Enter Burst Time for Process 2 -- 7
Enter Burst Time for Process 3 -- 3
```

OUTPUT:

PROCESS	BURST TIME	WAITING TIME	TURNAROUND TIME
P3	3	0	3
P0	6	3	9
P2	7	9	16
P1	8	16	24

Average Waiting Time -- 7.000000
Average Turnaround Time -- 13.000000

c)ROUND ROBIN CPU SCHEDULING ALGORITHM

PROGRAM:

```
#include<stdio.h>
main()
{
	int i,j,n,bu[10],wa[10],tat[10],t,ct[10],max;
	float awt=0,att=0,temp=0;
	clrscr();
	printf("Enter the no of processes -- ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\nEnter Burst Time for process %d -- ", i+1);
		scanf("%d",&bu[i]);
		ct[i]=bu[i];
	}
	printf("\nEnter the size of time slice -- ");
	scanf("%d",&t);
	max=bu[0];
	for(i=1;i<n;i++)
		if(max<bu[i])
			max=bu[i];
	for(j=0;j<(max/t)+1;j++)
		for(i=0;i<n;i++)
			if(bu[i]!=0)
				if(bu[i]<=t)
				 {
					tat[i]=temp+bu[i];
					temp=temp+bu[i];
					bu[i]=0;
				 }
				else
				 {
```

```
                                        bu[i]=bu[i]-t;
                                        temp=temp+t;
                                }
        for(i=0;i<n;i++)
        {
                wa[i]=tat[i]-ct[i];
                att+=tat[i];
                awt+=wa[i];
         }
        printf("\nThe Average Turnaround time is -- %f",att/n);
        printf("\nThe Average Waiting time is -- %f ",awt/n);
        printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND
        TIME\n");
        for(i=0;i<n;i++)
                printf("\t%d \t %d \t\t %d \t\t %d \n",i+1,ct[i],wa[i],tat[i]);
        getch();
}
```

INPUT:

Enter the no of processes – 3
Enter Burst Time for process 1 – 24
Enter Burst Time for process 2 -- 3
Enter Burst Time for process 3 -- 3
Enter the size of time slice – 3

OUTPUT:

The Average Turnaround time is – 15.666667
The Average Waiting time is -- 5.666667

PROCESS	BURST TIME	WAITING TIME	TURNAROUND TIME
1	24	6	30
2	3	4	7
3	3	7	10

d)PRIORITY CPU SCHEDULING ALGORITHM

PROGRAM :

```
#include<stdio.h>
main()
{
	int p[20],bt[20],pri[20], wt[20],tat[20],i, k, n, temp;
	float wtavg, tatavg;
	clrscr();
	printf("Enter the number of processes --- ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		p[i] = i;
		printf("Enter the Burst Time & Priority of Process %d --- ",i);
		scanf("%d %d",&bt[i], &pri[i]);
	}
	for(i=0;i<n;i++)
		for(k=i+1;k<n;k++)
			if(pri[i] > pri[k])
			{
				temp=p[i];
				p[i]=p[k];
				p[k]=temp;
				temp=bt[i];
				bt[i]=bt[k];
				bt[k]=temp;
				temp=pri[i];
				pri[i]=pri[k];
				pri[k]=temp;
			}
	wtavg = wt[0] = 0;
	tatavg = tat[0] = bt[0];
	for(i=1;i<n;i++)
	{
		wt[i] = wt[i-1] + bt[i-1];
		tat[i] = tat[i-1] + bt[i];
		wtavg = wtavg + wt[i];
```

```
            tatavg = tatavg + tat[i];
        }
        printf("\nPROCESS\t\tPRIORITY\tBURST TIME\tWAITING
        TIME\tTURNAROUND TIME");
        for(i=0;i<n;i++)
            printf("\n%d \t\t %d \t\t %d \t\t %d \t\t %d ",p[i],pri[i],bt[i],wt[i],tat[i]);
        printf("\nAverage Waiting Time is --- %f",wtavg/n);
        printf("\nAverage Turnaround Time is --- %f",tatavg/n);
        getch();
}
```

INPUT :

Enter the number of processes -- 5
Enter the Burst Time & Priority of Process 0 --- 10 3
Enter the Burst Time & Priority of Process 1 --- 1 1
Enter the Burst Time & Priority of Process 2 --- 2 4
Enter the Burst Time & Priority of Process 3 --- 1 5
Enter the Burst Time & Priority of Process 4 --- 5 2

OUTPUT :

PROCESS	PRIORITY	BURST TIME	WAITING TIME	TURNAROUND TIME
1	1	1	0	1
4	2	5	1	6
0	3	10	6	16
2	4	2	16	18
3	5	1	18	19

Average Waiting Time is --- 8.200000
Average Turnaround Time is --- 12.000000